PySpark Basic

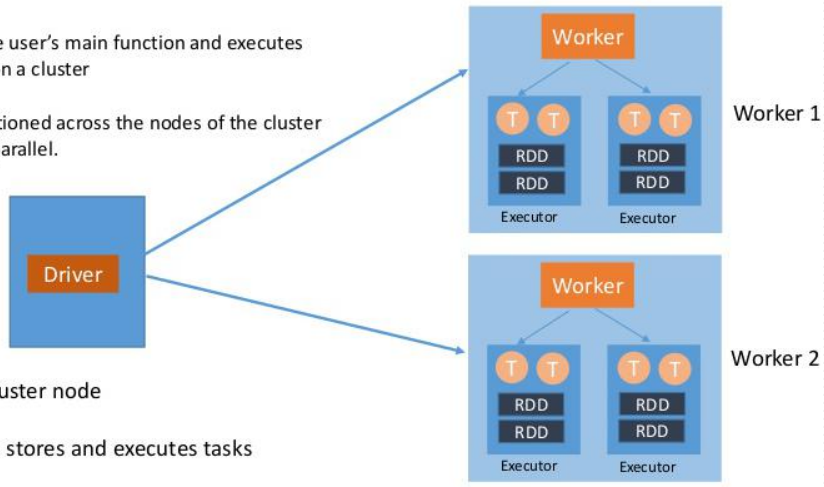
讲师：轩宇

PySpark

- Lesson Plan
- Installing Python Linux
- Spark Deployment
- Submit Application
- Uesr Action Analyser

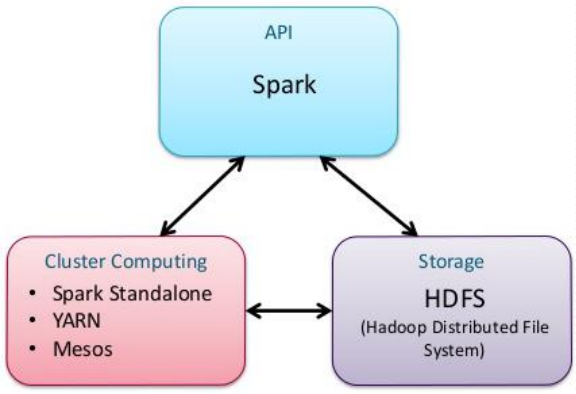
SPARK ARCHITECTURE

- Driver
  - Driver program that runs the user's main function and executes various parallel operations on a cluster
- RDDs
  - Collection of elements partitioned across the nodes of the cluster that can be operated on in parallel.

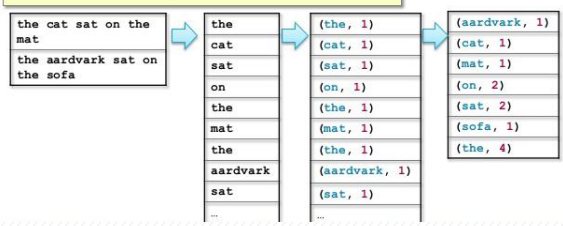


- Worker
  - Manages resources on cluster node
- Executor
  - Is the JVM process which stores and executes tasks
- Tasks
  - Executes the RDD operations

DISTRIBUTED PROCESSION WITH SPAK




```
> counts = sc.textFile(file) \
    .flatMap(lambda line: line.split()) \
    .map(lambda word: (word,1)) \
    .reduceByKey(lambda v1,v2: v1+v2)
```



SPARK V. HADOOP MAPREDUCE

- **Spark takes the concepts of MapReduce to the next level**
  - Higher level API = faster, easier development

```
> counts = sc.textFile(file) \
    .flatMap(lambda line: line.split()) \
    .map(lambda word: (word,1)) \
    .reduceByKey(lambda v1,v2: v1+v2)
> counts.saveAsTextFile(output)
```

```
public class WordCount {
  public static void main(String[] args) throws Exception {
    Job job = new Job();
     job.setJarByClass(WordCount.class);
     job.setJobName("Word Count");
     FileInputFormat.setInputPaths(job, new Path(args[0]));
     FileOutputFormat.setOutputPath(job, new Path(args[1]));
     job.setMapperClass(WordMapper.class);
     job.setReducerClass(SumReducer.class);
     job.setMapOutputKeyClass(Text.class);
     job.setMapOutputValueClass(IntWritable.class);
     job.setOutputKeyClass(Text.class);
     job.setOutputValueClass(IntWritable.class);
     boolean success = job.waitForCompletion(true);
     System.exit(success ? 0 : 1);
   }
}

public class WordMapper extends Mapper<LongWritable, Text, Text,
IntWritable> {
public void map(LongWritable key, Text value,
Context context) throws IOException, InterruptedException {
     String line = value.toString();
     for (String word : line.split("\\W+")) {
       if (word.length() > 0)
         context.write(new Text(word), new IntWritable(1));
       }
     }
   }
}

public class SumReducer extends Reducer<Text, IntWritable, Text,
IntWritable> {
  public void reduce(Text key, Iterable<IntWritable>
  values, Context context) throws IOException, InterruptedException {
    int wordCount = 0;
    for (IntWritable value : values) {
      wordCount += value.get();
    }
    context.write(key, new IntWritable(wordCount));
  }
}
```

PYSPARK BASICS